REMARKS

Claims 15-51, 66, 68-73, and 80-91 are pending. Claims 41-43, 45, 48, 50, 51, 66, and 69-73 are rejected. Applicants gratefully acknowledge the indication in the Office Action that claims 15-40 and 80-91 are allowed, and that claims 44, 46, 47, 49 and 68 include allowable subject matter and would be allowed if rewritten as suggested in the Office Action. Applicants respectfully request reconsideration of the claim rejections in view of the following remarks.

Telephone Conversation With Examiner

Examiner To is thanked for the telephone conversation conducted on January 29, 2009. Differences between the cited art and the claimed subject matter were discussed. Examiner stated that she will consider Applicants' remarks when examining the instant response.

Claim Rejections – 35 USC § 103

Claims 41-43, 45, 48, 51, 66, 69 and 70-72 are rejected under 35 U.S.C. §103(a) as being unpatentable over Inoue (US 5,760,789) in view of Hagemark et al (US 6,070,002). In addition, claim 50 is rejected under 35 U.S.C. §103(a) as being unpatentable over Inoue in view of Hagemark and further in view of Magar et al. (US 4,713,746). Applicants respectfully traverse the rejections and submit that at the very least, independent claims 41 and 66 are patentable over Inoue and Hagemark as follows.

For example, with regard to claim 41, the combination of Inoue and Hagemark does not disclose or fairly suggest, e.g., *one or more applications that generate tasks for processing by the coprocessor, . . .* and *a scheduler process for determining an order in which the tasks are processed, wherein the order accounts for any relative priority among a first application relating to a first set of tasks and one or more other applications relating to additional tasks* and *wherein the order accounts for a corresponding amount of processing time that the first application and the one or more other applications are entitled to*, as recited in claim 41.

In formulating the rejection of claim 41, the Office Action cites Col. 12, lines 5-17 of Inoue as disclosing a "scheduler process" that determines an order in which application tasks are processed, wherein the order accounts for "relative priority" and "amount of processing time", as claimed. However, it is respectfully asserted that reliance on Inoue in this regard is misplaced. While Inoue generally discloses a "scheduling means" for determining an order in which client (application) requests (for writing data) are sent to a graphic device for execution, Inoue merely discloses that the order in which the client application tasks are processed is based on the same order in which the client requests are received, and not on account of a "relative priority" and "amount of processing time", as claimed.

Inoue shows (in FIGs. 4 and 5) a display server that runs on a host system which supports a graphics display, keyboard and mouse, and which displays data by processing client (application) requests and hardware requests from the keyboard and mouse (see Col. 7, lines 50-64 and Col. 8, lines 22-28). Inoue shows in FIG. 9B, a scheduling process in which the order of processing display (writing) requests received by different client applications is based on the order in which the client requests are received, but that a display request by a hardware input device (e.g., mouse) will be executed ahead of other pending write requests of client applications that are received before the hardware display request, but not yet sent to the graphics processor for execution.

In particular, Inoue shows in FIG. 9B a sequence of graphics commands generated by a host in order of time (left side) and the order/sequence of the commands being executed (on the right side). FIG. 9B shows that the order of execution of commands related to client application requests is the same order in which the client commands are generated. However, a command to draw a font representing a mouse which is associated with a hardware display request is shown in FIG. 9B as being executed out of sequence, prior to any not-yet-executed graphics commands for requests received from the clients. (see, e.g., Col. 10, lines 52-59).

Inoue teaches that the execution order in FIG. 9B allows a mouse display request to be executed at substantially the time the display request is generated (see, e.g., Col. 10, lines 60-62) so that the displayed pointer will appear to a user to move at the same time that user moves the mouse, substantially in real time. This is in contrast to the conventional scheduling process depicted in FIG. 8B of Inoue in which an order of execution of display commands by clients and input hardware devices (e.g., mouse) is based solely on the order/time sequence at which the commands are generated, which can result in an undesirable delay between the time when the user moves a mouse and the time that the displayed cursor appears to the user to move on the display (see, Col. 4, lines 12-40).

Thus, in view of the above, Inoue teaches a scheduling order process (in both FIGs. 9B and 8B) in which client (application) display requests are scheduled in an order based on a "first come first serve basis", which clearly does not account for "relative priority" among applications, as claimed. Although Inoue discloses that preference is given for "out of sequence" execution of a hardware display request over pending client display requests, this preference/priority merely relates at most to "task priority" specific to hardware display requests over client display requests.

Moreover, in view of the above, there is seemingly nothing in Inoue that discloses or fairly suggests that the order in which application tasks are executed *accounts for a corresponding amount of processing time that the first application and the one or more other applications are entitled to,* as claimed in claim 41. While the cited section (Col. 12, lines 5-17) of Inoue discloses that a graphic device designates a writing instruction execution time corresponding to each writing request transferred thereto to set a time at which writing corresponding to the transferred writing request is to be executed on the display device, such "execution time" is clearly not a factor that accounts for the execution order as determined by the scheduling process, much less relate to an "amount of processing time" to which an application is entitled.

Furthermore, with regard to claim 41, the combination of Inoue and Hagemark does not disclose or fairly suggest, e.g., *applications that generate tasks for processing by the coprocessor, wherein the tasks are first stored in a user mode command buffer, and wherein said tasks are stored in a per-application context in said user mode command buffer,* as recited in claim 41.

In formulating the rejection of claim 41, the Office Action contends that Inoue discloses (in Col. 8, lines 50-58) that application tasks are first stored in a user mode command buffer, but acknowledges that Inoue does not teach that the tasks are stored in the user mode command buffer in a per-application context. Instead, the Office Action relies on Hagemark's teachings (in the Abstract, lines 3-4 and 18-21, Col. 3, lines 8-11, Col. 5, lines 38-41 and Col. 6, lines 4-10, as cited in the Office Action) to cure the admitted deficiencies of Inoue. However, these findings are respectfully traversed.

In the first instance, it is not entirely clear how Inoue discloses in Col. 8, lines 50-58 (FIG. 10) application tasks that are first stored in a *user mode command buffer*, as contended (without explanation) in Office Action. Indeed, while Inoue does disclose in FIG. 10 a *server command buffer* and a *graphics command buffer*, there is nothing in Inoue that specifically discloses or suggests that these command buffers are *user mode command buffers* as contemplated by the claimed inventions.

Moreover, while Hagemark may arguably disclose, in general, storing application tasks in a command buffer in a specific application context, Hagemark's teachings in this regard do not fairly support a finding of obviousness in combination with Inoue, for various reasons explained below.

Hagemark discloses (in the cited sections and, in particular, FIG. 4B) an application execution computing environment in which a user application (404A) establishes user application memory (424) in a user application memory portion (452) of a system memory (306) for exclusive use by the user application. The user application (404A) has an application context (420) that is associated with a pool of I/O buffers (426, 428) established by the application in the user application memory (424) and one or more converter contexts (430A, 430B) of a video imaging and compression module (VICM) established by the application in an OS driver memory portion (454) of the system memory (306). The buffer pools (426, 428) are used exclusively by the application (404A) in the application context (420) to control I/O of video data from/to a device, and the converter contexts (430A, 430B) are used exclusively by the application (404A) in the application context (420) to perform image conversion operations on real-time video data stored in the input buffer (426) and store results of the image conversion operation in the output buffer (428) (see, e.g., Col. 5, line 29, – Col. 6, line 22; and Claim 11).

Hagemark further discloses (in FIG. 7) that a converter context (704) includes an input command queue (706) and an output queue (708). The user application can use the converter context 704 to perform an operation by making a dmICSend call (which is part of the image conversion library 1820), wherein the dmICSend call (or a message encompassing the dmICSend call) is placed in the input command queue (706) and processed on a first-in first-out basis. (See Col. 8, lines 22-30; see also FIG 5.)

In view of the above, while Hagemark generally discloses an application that places application tasks in a command buffer in a specific application context, Hagemark teaches that such functionality is implemented to allow a user application to participate in and control/direct/manipulate processing within the application context, as opposed to delegating such control to hardware and/or the operating system. (Col. 10, lines 28-33). However, within the context of the claimed inventions, Hagemark does not disclose or suggest storing application tasks in user mode commend buffers in a per application context to allow a scheduler process to

determine an order in which the application tasks for various contexts are to be executed by a coprocessor.

In fact, in stark contrast, as noted above, Hagemark specifically discloses that application specific tasks are placed in the input command queue of a converter context by an application for execution on *a first-in first-out basis* within the application context. More specifically, Hagemark teaches a process in which an application stores application tasks in the command queue of a converter context that are to be executed by the converter context in the given application context and in the order in which such tasks are placed in the command queue by the application for execution.

The Office Action contends that it would have been obvious to combine the teachings of Inoue and Hagemark because Hagemark's teaching of storing tasks in a user mode command buffer in per application context would improve the Inoue system by expanding the ability of the user application to direct the image conversion processing and graphics processing (see page 3, paragraph 8 of the Office Action). However, it is respectfully submitted that this obviousness conclusion is not well founded

For example, the "motivation" proffered in the Office Action of using a user mode command buffer in per application context as taught by Hagemark to "expand the ability of the user applications" in the Inoue system to direct the image conversion processing and graphics processing, *actually teaches away* from the proposed combination of references. Indeed, in the context of the claimed subject matter, the process of initially storing application tasks in user mode command buffers in a per-application context manner allows a scheduler process (and not the applications) to determine an order in which the application tasks are processed by a coprocessor. This is in contrast to conventional systems in which applications can access a shared command buffer and place tasks in the buffer for execution by a coprocessor in a first-come-first-serve basis, whereby an application can "hog" use of the coprocessor by inputting a

large number of tasks into the buffer for execution. (see FIG. 1 and corresponding description in the current specification.).

Therefore, in contrast to Hagemark disclosed use of command buffers for applications to store application tasks in the associated application context for the purpose of expanding the ability of applications to more directly control processing of their application tasks, the claimed subject matter related to use of user mode command buffers to store application tasks per-application context to allow a scheduling process to determine an order of execution of application tasks stored in the buffers, actually limits the ability of applications to control the scheduling/execution of their processing application tasks. In this regard, the proffered "motivation" for combining Inoue and Hagemark is seemingly off point and does not reasonably support the finding of obviousness.

Furthermore, with regard to claim 66, for the same or similar reasons discussed above for claim 41, it is respectfully submitted that the combination of Inoue and Hagemark fails to disclose or fairly suggest, e.g., a coprocessor for use in connection with a coprocessing scheduler in which application tasks stored in a user mode command buffer in a per application context *are submitted to the coprocessor by a scheduler process that submits tasks to the coprocessor according to a priority of applications relating to said tasks . . .* and *wherein the priority determines the amount of coprocessor time one or more applications are entitled to*, as claimed in claim 66. Again, as demonstrated above, there is nothing in Inoue or Hagemark that discloses or suggests, e.g., submitting application tasks of applications to a coprocessor for execution in an order that is determined by a scheduler process based on application priority, as claimed.

In view of the above noted deficiencies of Inoue and Hagemark, it is respectfully submitted that independent claims 41 and 66 are patentable and allowable over the combination of Inoue and Hagemark. Moreover, claims 42-43, 45, 48, 50, 51, 69 and 70-72 are patentable over any combination of the cited art of record at least by virtue of their dependence from

respective base claims 41 and 66, for those reasons discussed above, as well as for patentably distinct elements contained in such claims. It is to be noted that Applicants generally deny, and do not concede to, any statement, position or averment in the Office Action in support of the claim rejections under 35 U.S.C. §103, which is not specifically addressed by the foregoing arguments and response. Withdrawal of the rejections under 35 U.S.C. § 103 is respectfully requested.

CONCLUSION

The Applicants believe that the present remarks are responsive to each of the points raised by the Examiner in the official action, and respectfully submit that all claims are in condition for allowance. Favorable consideration and passage to issue of the application at the Examiner's earliest convenience is earnestly solicited.

Date: February 19, 2009

/Joseph F. Oriti/
Joseph F. Oriti
Registration No. 47, 835

Woodcock Washburn LLP
Cira Centre
2929 Arch Street, 12th Floor
Philadelphia, PA 19104-2891
Telephone: (215) 568-3100
Facsimile: (215) 568-3439